443744

SP-1602/001/00

Automatic Paraphrasing in Essay Format

(SP Series)

SP-1602/001/00

Automatic Paraphrasing in Essay Format*

Sheldon Klein

July 21, 1964

SYSTEM DEVELOPMENT CORPORATION, SANTA MONICA, CALIFORNIA

*For presentation at the 1964 Annual Meeting of the Association for Machine Translation and Computational Linguistics, Indiana University, Bloomington, Indiana, July 29-30, 1964.

A-1108

Automatic Paraphrasing in Essay Format

Sheldon Klein
System Development Corporation
Santa Monica, California

1.0 Introduction

An operating program accepts as input an essay of up to 300 words in length, and yields as output an essay-type paraphrase that is a nonredundant summary of the content of the source text.

Although no transformations are used, the content of several sentences in the input text may be combined into a single sentence in the output. The format of the output essay may be varied by adjustment of program parameters. In addition, the system occasionally inserts subject or object pronouns in its paraphrases to avoid repetitious style.

The components of the system include a phrase structure and dependency parser, a routine for establishing dependency links across sentences, a program for generating coherent sentence paraphrases randomly with respect to order and repetition of source text subject matter, a control system for determining the logical sequence of the paraphrase sentences, and a routine for inserting pronouns.

The present version of the program requires that individual word class assignments be part of the information supplied with a source text, and also, that the grammatical structure of the sentences in the source conform to the limitations of the parsing system.

2.0 Dependency-Phrase Structure Parsing System

The parsing system used in the automatic essay writing experiments performed a phrase structure and dependency analysis simultaneously. Before describing its operation it will be useful to explain the operation of a typical phrase structure parsing system.

Cocke of IBM, Yorktown, developed a program for the recognition of all possible tree structures for a given sentence. The program requires a grammar of binary formulas for reference. While Cocke never wrote about the program himself, others have described its operation and constructed grammars to be used with the program.[1,2]

The operation of the system may be illustrated with a brief example. Let the grammar consist of the rules in Table 1; let the sentence to be parsed be:

A B C D

The grammar is scanned for a match with the first pair of entities occurring in the sentence. Rule 1 of Table 1, A + B = P, applies. Accordingly A and B may be linked together in a tree structure and their linking node labeled P.

(a) P

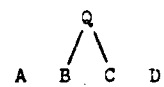
A B C D

But the next pair of elements, B + C, is also in Table 1. This demands the analysis of an additional tree structure.

1. A + B = P
2. B + C = Q
3. P + C = R
4. A + Q = S
5. S + D = T
6. R + D = U

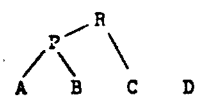
Table 1

Illustrative Rules for Cocke's Parsing System

(b) 

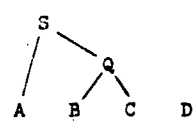
These two trees are now examined again. For tree (a), the sequence P + C is found in Table 1, yielding

(a)



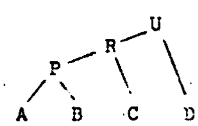
For tree (b), the pair A + Q is found in Table 1, but not the sequence Q + D. The result here is:

(b)



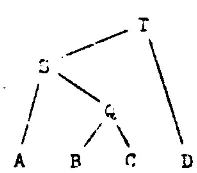
Further examination of tree (a) reveals that R + D is an entry in Table 1.

(a)



In tree (b), S + D is found to be in Table 1:

(b)



The analysis has yielded two possible tree structures for the

sentence, A B C D. Depending upon the grammar, analysis of longer sentences might yield hundreds or even thousands of alternate tree structures.

Alternatively, some of the separate tree structures might not lead to completion. If grammar rule 6 of Table 1, R + D = U, were deleted, the analysis of sentence (a) in the example could not be completed. Cocke's system performs all analyses in parallel and saves only those which can be completed.

The possibility of using a parsing grammar as a generation grammar is described in section 3.

2.1 Phrase Structure Parsing with Subscripted Rules

The phrase structure parsing system devised by the author makes use of a more complex type of grammatical formula. Although the implemented system does not yield more than one of the possible tree structures for a given sentence (multiple analyses are possible with program modification) it does contain a device that is an alternative to the temporary parallel analyses of trees that cannot be completed.

The grammar consists of a set of subscripted phrase structure formulas as for example, in Table 2. Here 'N' represents a noun or noun phrase class, 'V' a verb or verb phrase class, 'Prep' a preposition class, 'Mod' a prepositional phrase class, 'Adj' an adjective class, and 'S' a sentence class. The subscripts determine the order and limitations of application of these rules when generating as well as

1. $Art_0 + N_2 = N_3$
2. $Adj_0 + N_2 = N_3$
3. $N_1 + Mod_1 = N_1$
4. $V_1 + N_2 = V_2$
5. $Prep_0 + N_3 = Mod_1$
6. $N_3 + V_3 = S_1$

Table 2

Phrase Structure Rules

parsing. The use of the rules in parsing may be illustrated by example.

Consider the sentence:

'The fierce tigers in India eat meat.'

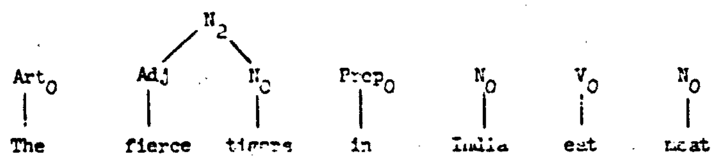
Assuming one has determined the individual parts of speech for each word:

Art_0	Adj_0	N_0	$Prep_0$	N_0	V_0	N_0
|	|	|	|	|	|	|
The	fierce	tigers	in	India	eat	meat

The parsing method requires that these grammar codes be examined in pairs to see if they occur in the left half of the rules of Table 2. If a pair of grammar codes in the sentence under analysis matches one of the rules and at the same time the subscripted of the components of the Table 2 pair are greater than or equal to those of the corresponding elements in the pair in the sentence, the latter pair may be connected by a single node in a tree, and that node labeled with the code in the right half of the rule in Table 2.

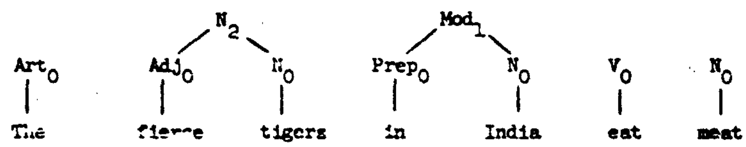
Going from left to right (one might start from either direction), the first pair of codes to be checked is $Art_0 + Adj_0$. This sequence does not occur in the left half of any rule.

The next pair of codes is $Adj_0 + N_0$. This pair matches the left half of rule 2 in Table 2, $Adj_0 + N_2 \rightarrow N_2$. Here the subscripts in the rule are greater than or equal to their counterparts in the sentence under analysis. Part of a tree may now be drawn.

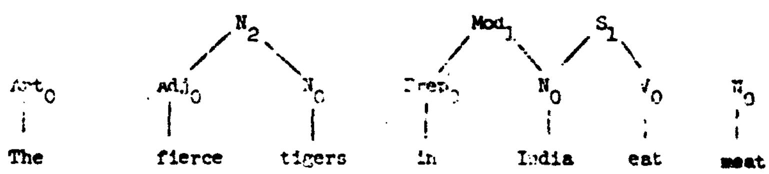


The next pair of codes to be searched for is N_0 + $Prep_0$. This is not to be found in Table 2.

The following pair, $Prep_0$ + N_0, fits rule 5, Table 2, $Prep_0$ + N_3 = Mod_1. The subscript rules are not violated, and accordingly, the sentence structure now appears as:



The next pair of codes, N_0 + V_0 also appears in Table 2, N_3 + V_3 = S_1. But if these two terms are united, the N_0 would be a member of two units. This is not permitted, e.g.,



When a code seems to be a member of more than one higher unit, the unit of minimal rank is the one selected. Rank is detemined

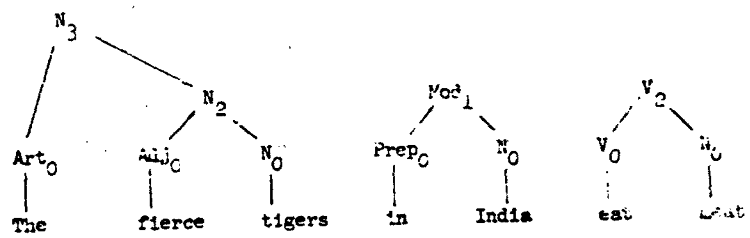
by the lowest subscript if the codes are identical. In this case, where they are not identical, S_1 (sentence) is always higher than a Mod_1 or any code other than another sentence type. Accordingly, the union of $N_0 + V_0$ is not performed. This particular device is an alternative to the temporary computation of an alternate tree structure that would have to be discarded at a later stage of analysis.

The next unit, $V_0 + N_0$, finds a match in rule 4 of Table 2, $V_1 + N_2 = V_2$, yielding:

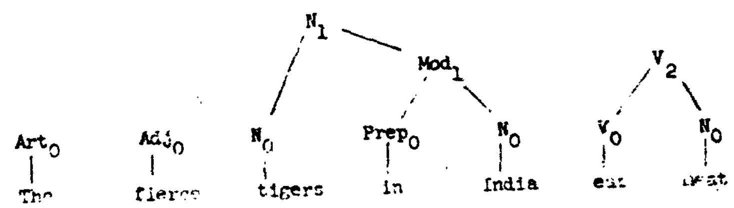
```
                N2                 Mod1          V2
               /  \               /    \        /  \
Art0       Adj0    N0        Prep0      N0    V0    N0
 |          |      |           |        |     |     |
The       fierce  tigers       in     India  eat   meat.
```

One complete pass has been made through the sentence. Successive passes are made until no new units are derived. On the second pass, the pair $Art_0 + Adj_0$, which has already been rejected, is not considered. However, a new pair, $Art_0 + N_2$, is now found in rule 1 of Table 2, $Art_0 + N_2 = N_3$.

The tree now appears as:

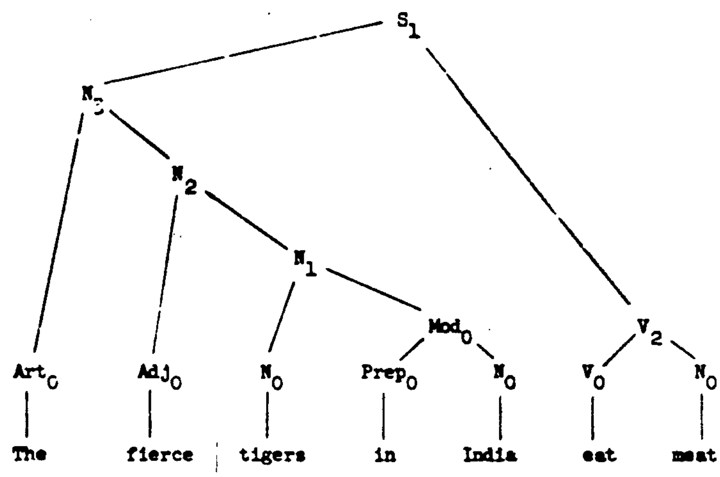


Continuing, the next pair accounted for by Table 2 is $N_0 + Mod_1$, which is within the domain of rule 3, $N_1 + Mod_1 = N_1$. Here the subscripts of the grammar rule are greater than or equal to those in the text entities. Now the N_0 associated with 'tiger,' is already linked to an Adj_0 unit to form an N_2 unit. However, the result of rule 3 in Table 2 is an N_1 unit. The lower subscript takes precedence; accordingly the N_2 unit and the N_3 unit of which it formed a part must be discarded, with the result:



On the balance of this scan through the sentence no new structures are encountered. A subsequent pass will link Adj_0 to N_1 producing an N_2 unit. Eventually this N_2 unit will be considered for linkage with V_2 to form a sentence, S_1, by rule 6 of Table 2. This linkage is rejected for reasons pertaining to rules of precedence.

A subsequent pass links Art_0 with this N_2 to form N_3 by rule 1 of Table 2. This N_3 is linked to V_2 by rule 6 of Table 2.



As the next pass yields no changes, the analysis is complete. This particular system, as already indicated, makes no provision for deriving several tree structures for a single sentence although it avoids the problem of temporarily carrying additional analyses which are later discarded.

2.2 Dependency

A phrase structure or immediate constituency analysis of a sentence may be viewed as a description of the relations among units of varied complexity. A dependency analysis is a description of relations among simple units, e.g., words. Descriptions of the formal properties of dependency trees and their relationship to immediate

constituency trees can be found in the work of David Hayes,[3] and Haim Gaifman.[4] For the purpose of this paper, the notion of dependency will be explained in terms of the information required by a dependency parsing program.

The particular system described next performs a phrase structure and dependency analysis simultaneously. The output of the program is a dependency tree superimposed upon a phrase structure tree.

Fundamentally, dependency may be defined as the relationship of an attribute to the head of the construction in which it occurs. In exocentric constructions, the head is specified by definition. Table 3 contains a set of grammatical rules which are sufficient for both phrase structure and dependency parsing. A symbol preceded by an asterisk is considered to be the head of that construction. Accordingly, in rule 1 of Table 3, $Art_0 + *N_2 = N_3$, the Art_0 unit is dependent on the N_2 unit. In rule 6 of Table 3, $*N_3 + V_3 = S_1$, the V_3 unit is dependent on the N_3 unit.

The method of performing a simultaneous phrase structure and dependency analysis is similar to the one described in the previous section. The additional feature is the cumulative computation of the dependency relations defined by the rules in the grammar. An example will be helpful in illustrating this point.

1. $Art_0 + {}^*N_2 = N_3$
2. $Adj_0 + {}^*N_2 = N_2$
3. ${}^*N_1 + Mod_1 = N_1$
4. ${}^*V_1 + N_2 = V_2$
5. ${}^*Prep_0 + N_3 = Mod_1$
6. ${}^*N_3 + V_3 = S_1$

Table 3

Dependency-Phrase Structure Rules

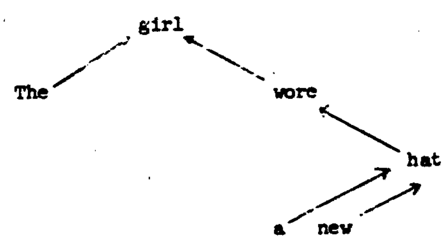
Consider the sentence:

'The girl wore a new hat.'

First the words in the sentence are numbered sequentially, and the word class assignments are made.

Art_0	N_0	V_0	Art_0	Adj_C	N_0
The	girl	wore	a	new	hat
<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>

The sequential numbering of the words is used in the designation of dependency relations. Looking ahead, the dependency tree that will be derived will be equivalent to the following:



where the arrows indicate the direction of dependency. Another way of indicating the same dependency analysis is the list fashion--each word being associated with the number of the word it is dependent on.

The	girl	wore	a	new	hat
<u>0</u>	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>
1		1	4	5	2

Consider the computation of this analysis. The first two units, $Art_0 + N_0$, are united by rule 1 of Table 3, $Art_0 + *N_0 = N_3$. The results will be indicated in a slightly different fashion than in the examples of section 2 1.

```
N3(1)____*N3(0)
  |         |
*Art0      *N0      *V0      *Art0     *Adj0     *N0
  |         |        |         |         |        |
 The       girl     wore       a        new      hat

  0         1        2         3         4        5
  1
```

All of the information concerning the constructions involving a particular word will appear in a column above that word. Each such word and the information above it will be called an entry. This particular mode of description represents the parsing as it takes place in the actual computer program.

The fact that Art + N_0 form a unit is marked by the occurrence of $*N_3$ at the top of entries 0 and 1. The asterisk preceding the N_3 at the top of entry 1 indicates that this entry is associated with

the head of the construction. The asterisks associated with the individual word tags indicate that at this level each word is the head of the construction containing it. This last feature is necessary because of certain design factors in the program.

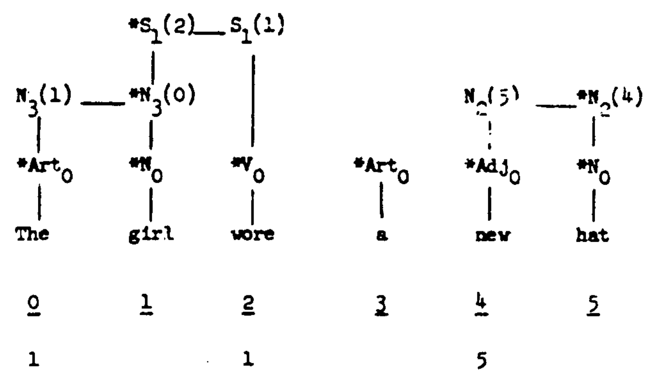
The numbers in brackets adjacent to the N_3 units indicate the respective partners in the construction. Thus the (1) at the top of entry 0 indicates that its partner is in entry 1, and the (0) at the top of entry 1, the converse. The absence of an asterisk at the top of entry 0 indicates that the number in brackets at the top of this entry also refers to the dependency of the English words involved in the construction; i.e., 'The' of entry 0 is dependent on 'girl' of entry 1. This notation actually makes redundant the use of lines to indicate tree structure. They are plotted only for clarity. Also redundant is the additional indication of dependency in list fashion at the bottom of each entry. This information is tabulated only for clarity.

The next pair of units accepted for by the program is $Adj_0 + N_0$. These, according to rule 2 of Table 3, are united to form an N_2 unit.

$N_3(1)$	$*N_3(0)$			$N_2(5)$	$*N_2(4)$
*Art	$*N_0$	$*V_0$	$*Art_0$	$*Adj_0$	$*N_0$
The	girl	wore	a	new	hat
0	1	2	3	4	5
1				5	

Here 'new' is dependent on 'hat.'

On the next pass through the sentence, the N_3 of entry 1, 'girl,' is linked to the V_0 of entry 2, 'wore,' to form an S_1 unit. It is worth noting that a unit not prefaced by an asterisk is ignored in the rest of the parsing.

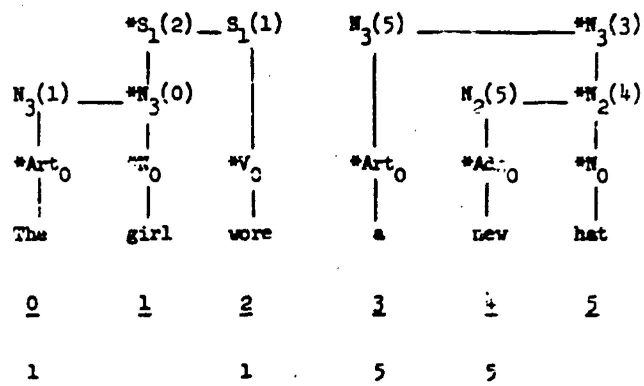


The new dependency emerging from this grouping is that of 'wore' upon 'girl.' The Art_0 of entry 3 plus the N_2 of entry 5 form the next unit combined, as indicated by rule 1 of Table 3. Note that the N_2 of entry 4 can be skipped because it is not preceded by an asterisk. Adjacent asterisked units are the only candidates for union.

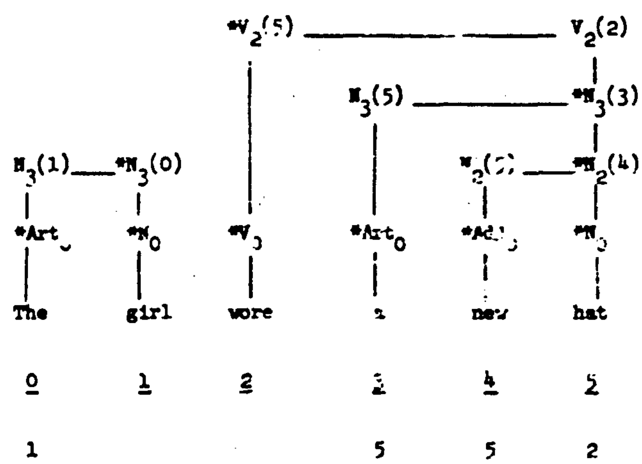
On the next pass through the sentence, the N_3 of entry 1, 'girl,' is linked to the V_0 of entry 2, 'wore,' to form an S_1 unit. It is worth noting that a unit not prefaced by an asterisk is ignored in the rest of the parsing.

```
              *S1(2)___S1(1)
                |        |
  N3(1) ____*N3(0)       |              N2(5) ____*N2(4)
    |           |        |                |          |
  *Art0        *N0      *V0     *Art0   *Adj0       *N0
    |           |        |        |       |          |
   The        girl     wore       a      new        hat

    0           1        2        3       4          5

    1                    1                5
```

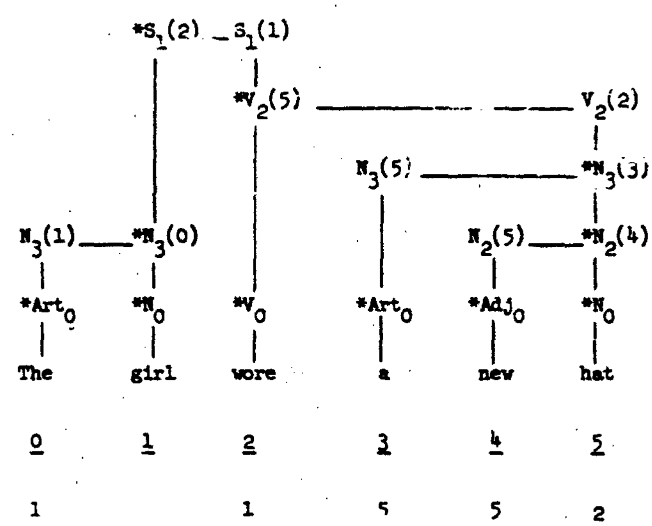
The new dependency emerging from this grouping is that of 'wore' upon 'girl.' The Art_0 of entry 3 plus the N_2 of entry 5 form the next unit combined, as indicated by rule 1 of Table 3. Note that the N_2 of entry 4 can be skipped because it is not preceded by an asterisk. Adjacent asterisked units are the only candidates for union.



On the next pass through the sentence, the V_0 of entry 2 is linked to the N_3 of entry 5 to form, according to rule 4 of Table 3, a V_2 unit. The S_1 unit, of which the V_0 is already a part, is deleted because the V_0 grouping takes precedence. The result is:



The next pass completes the analysis, by linking the N_3 of entry 1 with the V_2 of entry 2 by rule 6 of Table 3.



Note again that the dependency analysis may be read directly from the phrase structure tree; the bracketed digit associated with the top unasterisked phrase structure label for each entry indicates the dependency of the word in that entry.

The only entry having no unasterisked form at the top is 1. This implies that 'girl' is the head of the sentence. This choice of the main noun subject instead of the main verb as the sentence head is of significance in generating coherent discourse. The reasons for this are indicated in section 3.2.

3.0 Generation

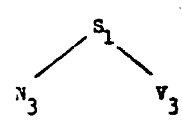
The discussion of generation is concerned with the production of both nonsensical and coherent discourse.

3.1 Grammatically Correct Nonsense

The generation of grammatically correct nonsense may be accomplished with the same type of phrase structure rules as in Table 2, 3 and 4. (The asterisks in Table 3 are not pertinent to generation.) A computer program implementating a phrase structure generation grammar of this sort has been built by Victor Yngve.[5]

The rules in Table 4 contain subscripts which, as in the parsing system, control their order of application. The rules may be viewed as rewrite instructions, except that the direction of rewriting is the reverse of that in the parsing system.

Starting with the symbol for sentence, S_1, $N_3 + V_3$ may be derived by rule 6 of Table 4.



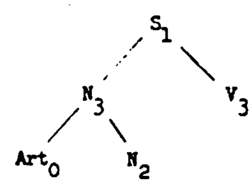
Note that a tree structure can be generated in tracing the history of the rewritings. Leftmost nodes are expanded first. The N_3 unit may be replaced by the left half of rule 1, 2 or 3. If the subscript of the N on the right half of these rules were greater than

1. $Art_0 + N_2 = N_3$
2. $Adj_0 + N_2 = N_2$
3. $N_1 + Mod_1 = N_1$
4. $V_1 + N_2 = V_2$
5. $Prep_0 + N_3 = Mod_1$
6. $N_3 + V_3 = S_1$
7. $N_0 = N_1$
8. $V_0 = V_1$

Table 4

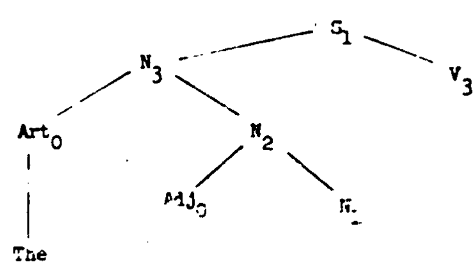
Illustrative Generation Grammar Rules

3, they would not be applicable. This is the reverse of the condition for applicability that pertained in the parsing system. Assume rule 1 of Table 4 is selected, yielding:



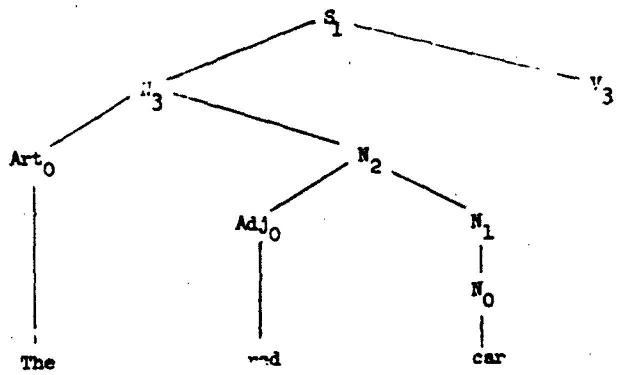
A node with a zero subscript cannot be further expanded. All that remains is to choose an article, at random, say 'the.' The N_2 unit can still be expanded. Note that rule 1 is no longer applicable because the subscript of the right hand member is greater than 1.

Suppose rule 2 of Table 4 is selected, yielding:

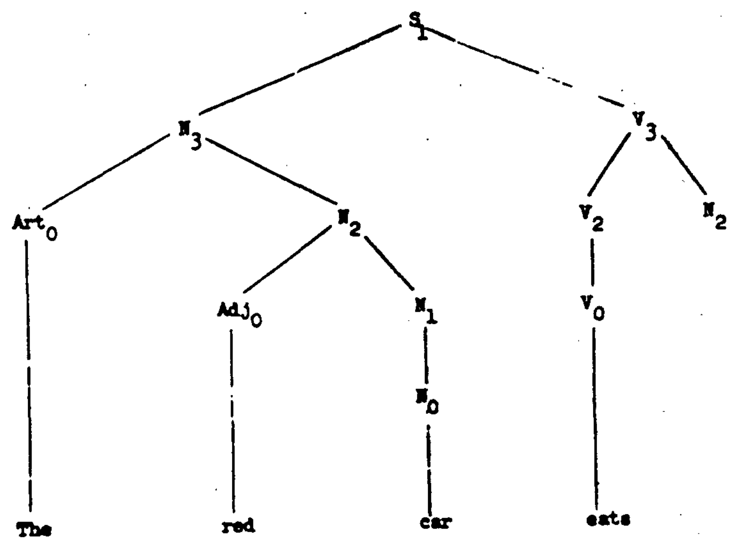


Now an adjective may be chosen at random, say, 'red.' The only expansions of N_1 are by rule 3 of Table 4, or rule 7, which makes it a terminal node. Note that rule 3 is recursive: that is, it may be used to rewrite a node repeatedly without reducing the value of the

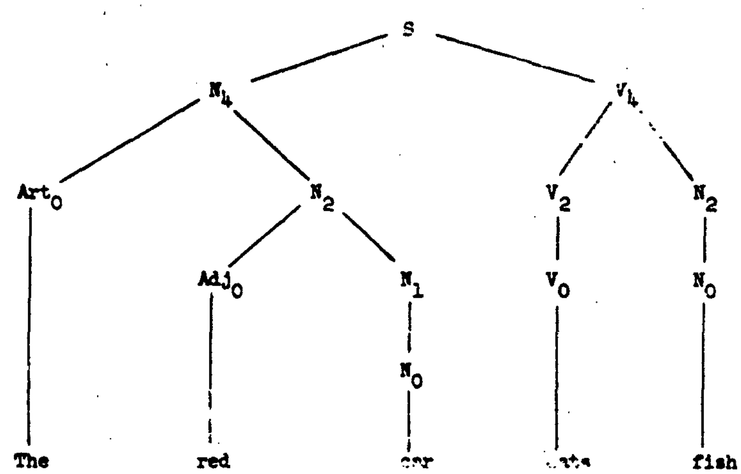
subscript. Accordingly an adjective string of indefinitely great length could be generated if rule 3 were chosen repeatedly. For the sake of brevity, next let rule 5 of Table 4 be selected. A noun may now be chosen at random. Say, 'car,' yielding:



Let the V_3 be written $V_1 + N_2$ by rule 4 of Table 4 and that V_1 rewritten as V_0 by rule 8 of Table 4. Let the verb chosen for this terminal node be 'eats.'


S1
N3
V3
Art0
N2
V2
N2
Adj0
N1
V0
N0
The
red
car
eats

The only remaining expandable node is N_2. Assume that N_0 is selected by rule 7. If the noun chosen for the terminal node is 'fish' the final result is:



With no restrictions placed upon the selection of vocabulary, no control over the semantic coherence of the terminal sentence is possible.

3.2 Coherent Discourse

The output of a phrase structure generation grammar can be limited to coherent discourse under certain conditions. If the vocabulary used is limited to that of some source text, and if it is required that the dependency relations in the output sentences not differ from those present in the source text, then the output sentences will be

coherent and will reflect the meaning of the source text. For the purpose of matching relations between source text and output text, dependency may be treated as transitive, except across prepositions other than 'of' and except across verbs other than forms of 'to be.'

A computer program which produces coherent sentence paraphrases by monitoring of dependency relations has been described elsewhere.[6,7] An example will illustrate its operation. Consider the text: 'The man rides a bicycle. The man is tall. A bicycle is a vehicle with wheels.'
Assume each word has a unique grammatical code assigned to it.

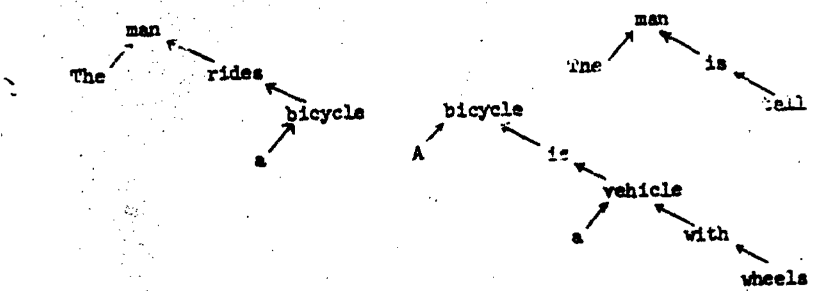
The	man	rides	a	bicycle
|	|	|	|	|
Art	N	V	Art	N

The	man	is	tall
|	|	|	|
Art	N	V	Adj

A	bicycle	is	a	vehicle	with	wheels
|	|	|	|	|	|	|
Art	N	V	Art	N	Prep	N

A dependency analysis of this text can be in the form of a network or a list structure. In either case, for purposes of

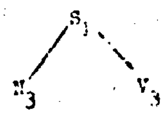
paraphrasing, two-way dependency links are assumed to exist between like tokens of the same noun. A network description would appear as follows:



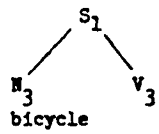
0	1	2	3	4	5	6	7	8
The	man	rides	a	bicycle.	The	man	is	tall.
1	0	1	4	2,10	6	1	6	7

9	10	11	12	13	14	15
A	bicycle	is	a	vehicle	with	wheels
10	4	10	13	11	13	14

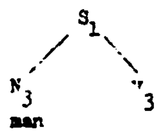
The paraphrasing program described would begin with the selection of a sentence type.



This generation program, in contrast with the method described above, chooses lexical items as soon as a new slot appears; for example, the main subject and verb of the sentence are selected now, while they are adjacent in the sentence tree. Assume that 'bicycle' is selected as the noun for N_3.

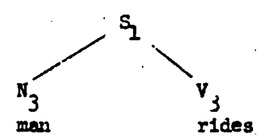


It is now necessary to find a verb directly or transitively dependent on 'bicycle.' Inspection of either the network or list representation of the text dependency analysis shows no verb dependent on 'bicycle.' the computer determines this by treating the dependency analysis as a maze in which it seeks a path between each verb token and the word 'bicycle.' Accordingly, the computer program requires that another noun be selected in its place; in this case, 'man.'

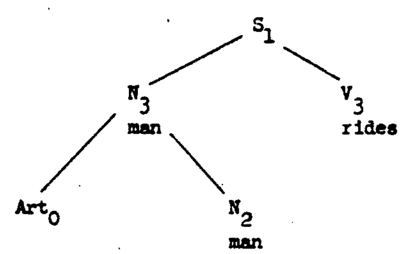


The program keeps track of which token of 'man' is selected.

It is now necessary to choose a verb dependent on 'man.' Let 'rides' be chosen.

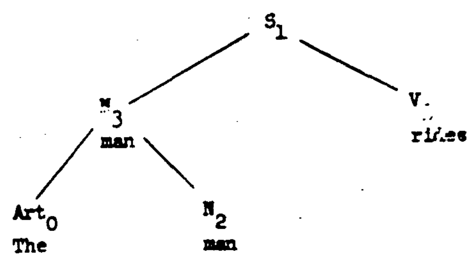


Now the N_3 may be expanded. Suppose rule 1 of Table 4 is chosen



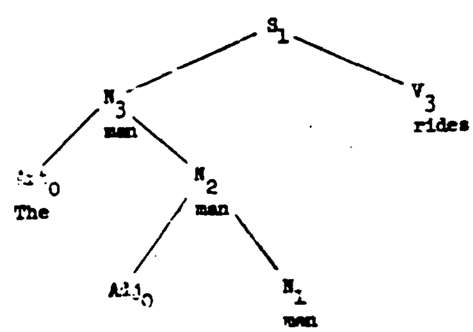
Note that 'man' is associated with the new noun phrase node, N_2.

It is now necessary to select an article dependent on 'man.' Assume 'a' is selected. While a path 'a' to 'man' does seem to exist in the dependency analysis, it crosses 'rides,' which is a member of a verb class treated as an intransitive link. Accordingly, 'a' is rejected. Either token of 'the' is acceptable, however. (Note that for simplicity of presentation no distinction among verb classes has been made in the rules of Tables 1 - 4.)

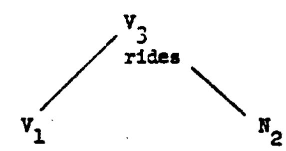


The 'Art$_0$' with a zero subscript cannot be further expanded.

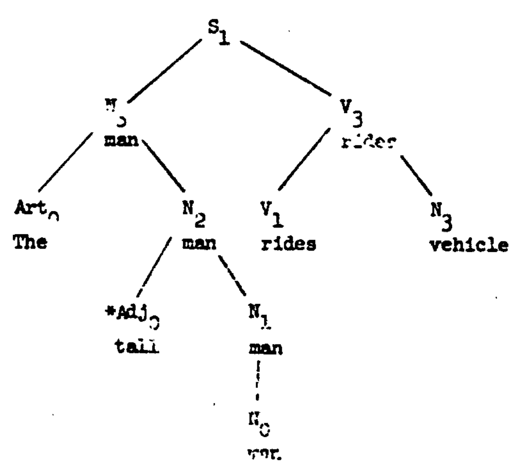
Let the 'N_2' be expanded by rule 2 of Table 4.



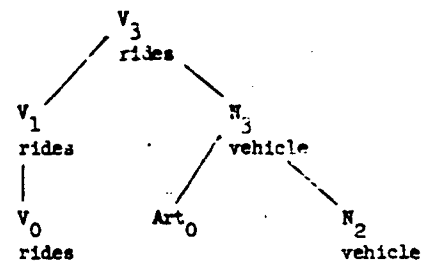
Let 'N_0' be chosen as the next expansion of N_1, by rule 7. Now the only node that remains to be expanded is V_3. If rule 4 of Table 4 is chosen, the part of the tree pertinent to 'rides' becomes:



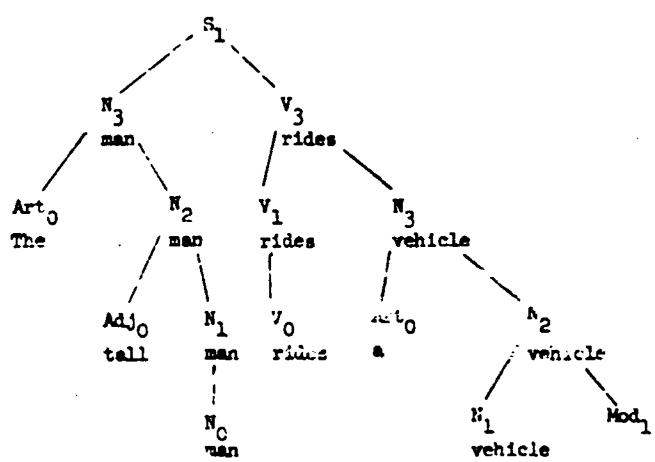
A noun dependent on 'rides' must now be found. Either token of 'man' would be rejected. If 'vehicle' is chosen, a path does exist that traverses a transitive verb 'is' and two tokens of 'bicycle.'



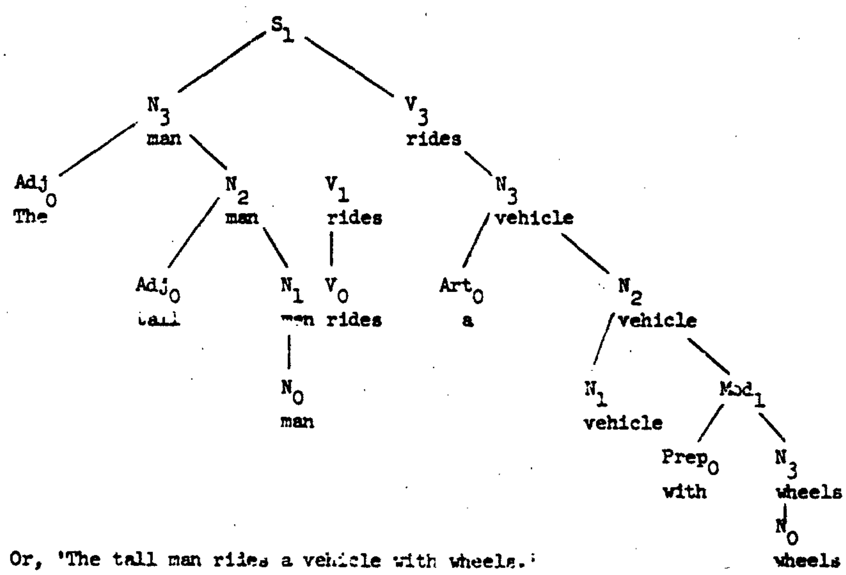
Let 'V_0' be chosen as the rewirting of V_2 by rule 8 of Table 4, and let the N_3 be rewritten by rule 1 of Table 4. The pertinent part of the tree now appears as follows:



Assume that 'a' is chosen at the article and that N_2 is rewritten as N_1 + Mod_1 by rule 3 of Table 4. The result is:



The Mod_1 is purely a slot marker, and no vocabulary item is selected for it. If the Mod_1 is rewritten $Prep_0 + N_3$ by rule 5 of Table 4, 'with' would be selected as a preposition dependent on 'vehicle,' and 'wheels' as a noun dependent on 'with.' After the application of rule 7, the N_3 would be rewritten N_0, completing the generation.



Or, 'The tall man rides a vehicle with wheels.'

In cases where no word with the required dependencies can be found, the program in some instances deletes the pertinent portion of the tree, in others, completely aborts the generation process. The selection of both vocabulary items and structural formulas is done randomly.

4.0 An Essay Writing System

Three computer programs were described in sections 2 and 3. The first performs a unique dependency and phrase structure analysis of individual sentences in written English text, the vocabulary of which has received unique grammar codes. The power of this program is limited to the capabilities of an extremely small recognition grammar.

The second program generates grammatically correct sentences without control of meaning. The third program consists of a version of the second program coupled with a dependency monitoring system that requires the output sentences to preserve the transitive dependency relations existing in a source text. A unique dependency analysis covering relations both within and among text sentences is provided as part of the input. The outputs of this third program are grammatically correct, coherent paraphrases of the input text which, however, are random with respect to sequence and repetition of source text content.

What is called an 'essay' writing system in this section consists of all the programs described earlier, plus a routine for assigning dependency relations across sentences in an input text and a routine which insures that the paraphrase sentences will appear in a logical sequence and will not be repetitious with respect to the source text content. Still another device is a routine that permits the generation of a paraphrase around an outline supplied with a larger body of text. In addition, several generative devices have been added: routines

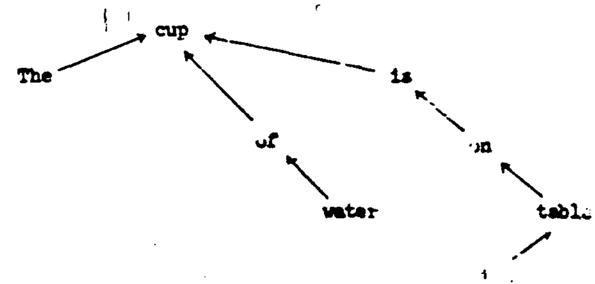
for using subject and object pronouns even though none occur in the input text, routines for generating relative clauses, although, again, none may occur in the input text, and a routine for converting source text verbs to output text forms ending in '-ing.'

4.1 Dependency Analysis of an Entire Discourse

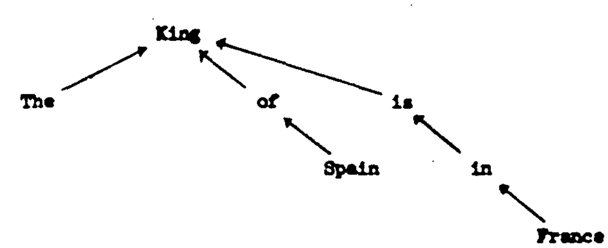
After the operation of the routine that performs a dependency and phrase structure analysis of individual sentences, it is necessary for another program to analyze the text as a unit to assign dependency links across sentences and to alter some dependency relations for the sake of coherent paraphrasing. The present version of the program assigns two-way dependency links between like tokens of the same noun. A future version will be more restrictive and assign such links only among tokens having either similar quantifiers, determiners, or subordinate clauses, or which are determined to be equatable by special semantic rules. This is necessary to insure that each token of the same noun has the same real world referent.

While simple dependency relations are sufficient for paraphrasing the artificially constructed texts used in the experiments described in this paper, paraphrasing of unrestricted English text would demand special rule revisions with respect to the direction and uniqueness of the dependency relation. The reason for this is easily understood by a simple example familiar to transformationalists.

'The cup of water is on the table'



'The King of Spain is in France'



The parsing system would yield the same type of analysis for each sentence. Yet it would be desirable to be able to paraphrase the first sentence with:

'The water is on the table'

without the possibility of paraphrasing the second sentence with

'Spain is in France.'

Accordingly, a future modification of the routine described in this section would, after noting the special word classes involved, assign two-way dependency links between 'cup' and 'of' and also between 'of' and 'water,' but take no such action with words 'King,' 'of,' and 'Spain' in the second sentence. This reparsing of a parsing has significance for a theory of grammar, and its implications with respect to stratificational and transformational models is discussed in section 5.

4.2 Paraphrase Formatting

Control over sequence and nonrepetition of the paraphrase sentences is obtained through the selection of an essay format. The format used in the experiments performed consists of a set of paragraphs each of which contains only sentences with the same main subject. The ordering of the paragraphs is determined by the sequence of nouns as they occur in the source text. The ordering of sentences within each paragraph is partially controlled by the sequence of verbs as they occur in that text.

Before the paraphrasing is begun, two word lists are compiled by a subroutine. The first list contains a token of each source text noun that is not dependent on any noun or noun token occurring before it in the text. The tokens are arranged in source text order. The second list consists of every token of every verb in the text, in sequence.

The first noun on the list is automatically selected as the main subject noun for each sentence that is to be generated. As many generations are attempted as there are verbs on the verb list. The main verb for each such sentence generation attempt is taken in sequence from those on the list. Once a sentence is successfully generated, the token of the verb used is deleted from the verb list. Nonsequential use of verbs can occur in relative clauses or modifying phrases. In these instances also, the verbs or verb stem tokens used are deleted from the verb list. When every verb on the list has been tried as the main verb for a particular main subject noun, a new paragraph is begun and the next noun on the list becomes the main subject for each sentence. The process is continued until the noun list is exhausted. It may happen that some nouns do not appear as subjects of paragraphs even though they appear on the noun list, because they do not occur as main subjects in the source text. (This procedure was arbitrarily selected as suitable for testing the program; other formats for essay generation can be implemented.)

The use of an outline as the basis for generating an essay from a larger body of text is accomplished simply; the boundary between the outline and the main body of text that follows is marked. The noun list is limited only to those nouns occurring in the outline. The verbs selected still include those in the main text as well as the ones in the outline. Theoretically, the main text could consist of a large library; in that case the outline might be viewed as an

information retrieval request. The output would be an essay limited to the subject matter of the outline but drawn from a corpus indefinitely large in both size and range of subject matter.

4.3 Generation of Word Forms Not Present in the Source Text

Earlier experiments indicated that in many instances reasonable paraphrases could be performed with the method described herein if the dependency relations held only among stems rather than among full word forms and if the stems were subsequently converted to forms of the proper grammatical category. The present system will accept a verb form with proper dependency relations and use it in a form ending in '-ing' when appropriate.

Relative clauses may be generated even though no relative pronouns occur in the source text. Where the generation process requires a relative pronoun, 'who' or 'which' is inserted into the proper slot depending on the gender of the appropriate antecedent. All the descriptors of the antecedent are then assigned to the relative pronoun. As far as the operation of all programs is concerned, the pronoun is its antecedent. Accordingly, if a routine is to inquire whether a particular verb is dependent on a relative pronoun, the request is formulated in terms of the verb's dependency on the antecedent of the relative pronoun.

The system may also generate subject and object pronouns although such forms do not occur in the source text. The use of subject and object pronouns is accomplished by separate routines. Subject pronouns

may be used randomly at a frequency that may be controlled by input parameters. After the occurrence of the first sentence in a paragraph, a subject pronoun of appropriate gender and number may be used as the main subject of subsequent sentences within the paragraph if program generated random numbers fall within a specified range.

The occurrence of an object pronoun of appropriate number and gender is obligatory whenever a nonsubject noun would normally be identical with the last nonmain subject noun used. A special storage unit containing the last nonmain subject noun used gives the program easy recognition of the need for a pronoun.

4.4 Computer Generated Essays

A number of essays were produced from varied texts, all of which were specially constructed so as to be suitable for parsing by a small dependency and phrase structure grammar. The parsing recognition grammar is contained in Table 5. (Because the material covered forms a related whole, Table 5 and all subsequent tables are gathered in an appendix at the end of this document.) The generation grammar is shown in Table 6. The recognition grammar is more powerful than the generation grammar. The first input text made no use of an outline; more exactly, because the program anticipates the presence of an outline, the entire text was its own outline. Input Text I is contained in Table 7, part 1. Its essay paraphrase, Output Text I, is contained in Table 7, part 2. Note that the generation rules used in producing Output Text I do not contain the rule for producing forms ending in '-ing.'

The use of this rule and the associated device for converting verb forms ending in '-ing' is illustrated in Output Texts III and IV, which appear in Tables 10 and 11.

Unambiguous word class assignments were part of the input data. As an example, the first sentence of Input Text I, Table 7, was coded:

Clever (adj.) John (noun, masc., sg.) met (verb, 3rd pers. sg.)
Mary (noun, fem., sg.) in (prep.) the (art) park (noun, neut. sg.).

Capital letters were indicated by a '+' sign preceding the first letter or word because a computer does not normally recognize such forms. The presence of an initial capital letter with a word coded 'noun' provided the program with information sufficient to distinguish such forms as belonging to a separate class. Two verb classes were distinguished in the recognition grammar, forms of 'to be' and all others; also, two preposition classes were established, 'of' and all others. Ad hoc word class assignments were made in the case of 'married' in Input Text I, Table I, which was treated as a noun, and the case of 'flamenco' in Input Text II, Table 9, which was labeled an adjective. In each case this was done in order to avoid a more complicated generation grammar. A price was paid for this simplification as can be seen in the phrase 'Flamenco Helen' generated in Output Text II, Table 9. The uncapitalized form of 'bentley' which appears in several

of the later paraphrases is not a typographical error, but rather is intended to reflect the use of capitalization to distinguish a separate word class. In order not to assign 'bentley' to the same class as 'John' it was left uncapitalized. (The device is not wholly adequate.) The noun classes differentiated by the presence or absence of prefixed '+' were manipulated directly within the program rather than by special rules for each class. The program prevented a form prefixed by a '+' from taking an article and from being followed by a form ending in '-ing.'

It should be noted that the spacing of the output texts in Table 7 and beyond is edited with respect to spacing within paragraphs. Only the spacing between paragraphs is similar to that of the original output.

Table 8 contains an essay paraphrase generated with the requirement that only the converse of Input Text I dependencies be present in the output.

5.0 Discussion

There are several comments that can be made about the essay writing program with respect both to the functioning of the programs and to the implications for linguistic theory suggested by the results.

5.1 Program

The compiled program occupies about 12,000 registers of Philco 2000 core storage, approximately 8,000 registers of which are devoted to tables. The JOVIAL program contains approximately 750 statements. Because of space limitations, the largest text the system can paraphrase is 300 English words, counting periods as words.

One early version of the system took an hour and a half to paraphrase 150 words of text; various attempts were made to control this processing time. Two programming devices used in this effort are described below.

Because the generation process involves a search of a network--the dependency structure of the text--the processing time would be expected to increase exponentially with text size. The ... factors that control the exponential rate of growth, besides text length, are the amount of connectivity among words and the syntactic complexity required of the sentences generated. Text that seldom repeats tokens of nouns would yield a nearly linear network, and the exponential increase of processing time per word with respect to length would not ... noticeable for short texts. However, the texts paraphrased in this paper had a fairly high frequency of repetition of noun tokens. The network representing the dependencies was made relatively linear by having the program link a noun token only to its immediately preceding token. Because dependency is transitive, all computed results were the same as if each token of a noun were linked to every other token of the same noun. Because of this linking convention, the dependency network was sufficiently linear to ke... the rate of increase linear with respect to text length, at least for the examples used in this paper.

Another device contributing to the reduction of processing time is tree pruning. The program generates a tree. If a subconstruction is initiated that cannot be carried to completion, it is often deleted without abandonment of the remainder of the generation tree. Unrealizable adjectives are among the units pruned. The addition of a routine to prune modifying phrases reduced the processing time to approximately 10% of the time required without the routine when the system was set to favor text with numerous modifying phrases.

The average time for generating an essay from an input of about 150 words is now 7 to 15 minutes, depending on the syntactic complexity required of the output. The processing time for producing a text from a 50-word source is about 15 minutes. From these figures it can be seen that the processing time per word increases linearly with the length of the text--15 seconds per word for a 50-word text input, about 45 seconds a word for a 150-word text input.

5.2 Theoretical Implications

The present version of the automatic essay writing system could not operate satisfactorily with unrestricted English text as input. For it to do so would require refinement of the dependency analysis, which was derived from immediate constituency considerations. As indicated earlier, reassignment of dependency links on the basis of the presence of numerous special word classes would be necessary. The problem presented by the necessity for recognizing multiple parsings of English sentences remains as another major hurdle.

The fact that verbs having appropriate dependency relations in source texts were satisfactorily used as '-ing' forms in paraphrases suggests a more general system in which input text words belonging to a variety of grammatical classes could be converted to new forms in output text by the appropriate application of what might be described as inflectional and derivational processes.

Such a system would have significance for linguistic theory. Even the system described earlier accomplishes the work of a number of transformations without using any. While a transformational grammar might be used to produce paraphrases beyond the scope of this system, the work of many transformations can be accomplished with a simpler conceptual framework. A transformation is an empirical generalization about a relationship between strings. In contrast, a transitive interpretation of dependency relations can often be used to predict the relationship a transformation represents.

Even though the essay writer described in this paper is an applied system, any complete theory of grammar should be able to account for its operation. I do not believe that transformational theory can do so.

A stratificational model of language might have more explanatory power. If, as in Sydney Lamb's model,[8,9] one posits the existence of a sememic stratum above a lexemic one, an explanation can be provided. Dependency relations may be viewed as a lexemic counterpart of tactic

relations among sememes. A dependency structure defining relations among lexemic units would have many very similar counterparts on the sememic stratum, somewhat as a listing of allomorphs in a language might resemble a listing of morphemes. The experiments described operated under conditions where the dependency structure was a close approximation to the semotactic structure which is posited as being the proper domain for manipulating meaning relations between one text and another. The first dependency analysis is analagous to lexotactic analysis. A refinement of this analysis might correspond to a semotactic analysis. Conceivably, a sufficiently refined system might come to resemble a dynamic implementation of a stratificational model.

References

1. Hays, D. G. Automatic Language Data Processing. In H. Borko (Ed.), Computer Applications in the Behavioral Sciences. Englewood Cliffs, N. J.: Prentice-Hall, 1962.

2. Robinson, Jane. Preliminary Codes and Rules for the Automatic Parsing of English. Memorandum RM-3339-PR, The RAND Corporation, Santa Monica, California, 1962.

3. Hays, D. G. Grouping and Dependency Theories. In H. P. Edmundson (Ed.), Proceedings of the National Symposium on Machine Translation. Englewood Cliffs, N. J.: Prentice-Hall, 1961.

4. Gaifman, H. Dependency Systems and Phrase Structure Systems. Memorandum P-2315, The RAND Corporation, Santa Monica, California, 1961.

5. Yngve, H. H. A Model and an Hypothesis for Language Structure. Proceedings of the American Philosophical Society, 1960, 104, 444-446.

6. Klein, S. Automatic Decoding of Written English. Ph.D. Dissertation in Linguistics, University of California, Berkeley, June 1963.

7. Klein, S., and Simmons, R. F. Syntactic Dependence and the Computer Generation of Coherent Discourse. Mechanical Translation, 1963, 7(2), 50-61.

8. Lamb, S. The Sememic Approach to Structural Semantics. In Kimbel A. Romney and Roy D'Andrede (Eds.), Transcultural Studies in Cognition. American Anthropologist, 1964, 66(3), Part 2.

9. White, J. The Methodology of Sememic Analysis with Special Application to the English Preposition. Mechanical Translation (in press).

Appendix

1. $Adj_0 + *N_3 = N_3$
2. $Adv_0 + *V_1 = V_2$
3. $*N_1 + SbCn_1 = N_2$
4. $*N_1 + Mod_1 = N_2$
5. $*N_4 + V_3 = S_1$
6. $*V_2 + N_4 = V_3$
7. $*V_2 + Mod_1 = V_3$
8. $*V\text{-}is_1 + Adj_0 = V_3$
9. $*V\text{-}is_1 + N_4 = V_3$
10. $*V\text{-}is_1 + Mod_1 = V_3$
11. $Art_0 + *N_3 = N_3$
12. $*Prep_0 + N_4 = Mod_1$
13. $*Pn\ Rcn_0 + N_3 = SbCon_1$
14. $*Part_0 + N_4 = Mod_1$

Table 5

Recognition Grammar

1. $Art_0 + N_1 = N_2$
2. $Adj_0 + N_1 = N_1$
3. $N_2 + SbCon_1 = N_3$
4. $N_2 + Mod_1 = N_4$
5. $N_0 = N_1$
6. $V_1 + N_4 = V_2$
7. $V_0 = V_1$
8. $Part_0 + N_3 = Mod_1$
9. $Prep_0 + N_3 = Mod_1$
10. $N_4 + V_4 = S_1$
11. [illegible]$Ren_0 + V_2 = SbCn_1$

Table 6

Generation Grammar

Clever John met Mary in the park. John married Mary. Mary loved John. Mary wanted a child. Mary had a child. Mary raised a child. John was a successful businessman who worked for a corporation. Mary was penniless. John secretly loved Helen who was beautiful. Helen who also loved John was married to Peter. Mary was a friend of Helen. Peter was a buddy of John. Helen who was friendly often ate lunch with Mary. John played golf with Peter. John wanted Helen. Helen wanted John. Divorce was impossible. The solution was simple. John liked Mary. Helen liked Peter. John killed Peter. Helen killed Mary. The end was happy.

Table 7, Part 1

Input Text I

John who married penniless Mary met her. Clever John was a businessman. He loved friendly Helen. He played golf. He wanted Helen. John who killed a buddy liked penniless Mary.

Mary in the park who wanted a child loved clever John. She had a child. She raised it. She was a friend of friendly beautiful Helen.

Beautiful Helen loved successful John. Beautiful Helen was married. Helen who wanted John ate lunch. She liked a buddy. She killed Mary.

Peter was a buddy.

Table 7, Part 2

Output Text I

John loved Mary. John loved Helen. He wanted her.

Mary who married John met him. Mary who killed Helen liked John.

Child wanted Mary. It had her. It raised her.

Helen loved John. She wanted him.

Peter who killed him liked Helen.

Lunch ate her.

Golf played John of Peter.

Table 8

Paraphrase of Input Text I Using Converse of Dependencies

(Outline)

Clever John met Mary in the park. John married Mary. Mary loved John. Mary wanted a child. Mary had a child. Mary raised a child. John was a successful businessman who worked for a corporation. Mary was penniless. John secretly loved Helen who was beautiful. Helen who also loved John was married to Peter. Mary was a friend of Helen. Peter was a buddy of John. Helen who was friendly often ate lunch with Mary. John played golf with Peter. John wanted Helen. Helen wanted John. Divorce was impossible. The solution was simple, John liked Mary. Helen liked Peter. John killed Peter. Helen killed Mary. The end was happy.

(Main Text)

A businessman is a man who likes money. John was a gangster. Peter was a bullfighter. Mary was a countess. Helen was a flamenco dancer. Lunch is a midday meal. A gangster commits crimes. A bullfighter fights bulls. Bulls are dangerous animals. The gangster drives a bentley. The flamenco dancer has many admirers. The countess owns a castle.

Table 9, Part 1

Input Text II

John who married penniless Mary met her. Clever John who commits crimes was a businessman. Clever John who drives a bentley loved a flamenco dancer. John played golf. He wanted Helen. Clever John who killed Peter liked Mary. John who likes money is a man. Clever John was a gangster.

Mary loved a successful businessman. Mary who was a countess wanted a child. Penniless Mary had it. Penniless Mary raised it. She was a friend. Mary in the park owns a castle.

Flamenco helen loved clever John. She was married. She ate lunch with Mary. Helen wanted John. She liked Peter. Helen killed a countess. Helen who has many admirers was a dancer.

Peter who fights bulls was a buddy of John. He was a bullfighter.

Table 9, Part 2

Output Text [illegible]

(Outline)	The hero is Peter. The unfaithful husband is John who commits murder.
(Main text)	John was a gangster. The gangster drives a bentley. A gangster commits crimes. John was a successful businessman who works for a corporation. Bulls are dangerous animals. Peter was a bullfighter. A bullfighter fights bulls.

Table 10, Part 1

Input Text III

A hero fighting bulls is Peter. He was a bullfighter.

The husband committing murder is successful John who was a gangster driving a bentley. A husband commits crimes. The successful unfaithful husband is a successful businessman.

Table 10, Part 2

Output Text III

With Conversion of Source Text Verbs to Forms in '-ing'

(Outline) The hero is Peter. The homewrecker is Helen. The unfaithful husband is John who commits murder The Fool housewife is Mary.

(Main text) John is a successful businessman who works for a corporation. A businessman is a man who likes money. John was a gangster. Peter was a bullfighter. Mary was a countess. Helen was a dancer. A gangster commits crimes. A bullfighter fights bulls. Bulls are dangerous animals. The gangster drives a bentley. The dancer has many admirers. The dancer wears a hat. The countess owns a castle. John secretly loved Helen who was beautiful. Helen who also loved John was married to Peter. John wanted Helen. Helen wanted John. Divorce was impossible. The solution was simple. John killed Peter. Helen killed Mary. The end was happy.

Table 11, Part 1

Input Text IV

A hero fighting bulls is Peter. He was a bullfighter.

The beautiful homewrecker who wanted a gangster who commits crimes is Helen. The homewrecker was a dancer who has many admirers. She wears a hat. She loved successful John who loved the dancer. A beautiful homewrecker was married. She killed Mary who owns a castle.

An unfaithful husband liking money is the gangster driving a bentley. He commits murder. The unfaithful husband working is a successful businessman. He is a man. The husband was a gangster. The unfaithful husband wanted Helen. The husband killed Peter.

Table 11, Part 2

Output Text IV

With Conversion of Verbs to Forms Ending in "-ing"

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
AUTOMATIC PARAPHRASING IN ESSAY FORMAT.
Scientific rept., SP-1602/001/00, by
S. Klein. 21 July 1964, 60p., 11 tables
Unclassified report

DESCRIPTORS: Language.

Describes an operating computer program that accepts as input an essay of up to 300 words in length, and yields as output an essay-type paraphrase that is a nonredundant summary of the content of

UNCLASSIFIED

UNCLASSIFIED

the source text. Reports that although no transformations are used, the content of several sentences in the input text may be combined into a sentence in the output. Further reports that the format of the output essay may be varied by adjustment of program parameters, and that the system occasionally inserts subject or object pronouns in its paraphrases to avoid repetitious style.

UNCLASSIFIED